Computer Science and Artificial Intelligence Laboratory

Technical Report

MIT-CSAIL-TR-2008-018 April 8, 2008

ZigZag Decoding: Combating Hidden Terminals in Wireless Networks

Shyamnath Gollakota and Dina Katabi

massachusetts institute of technology, cambridge, ma 02139 usa — www.csail.mit.edu

ZigZag Decoding: Combating Hidden Terminals in Wireless Networks

Shyamnath Gollakota Dina Katabi
gshyam@mit.edu dk@mit.edu

ABSTRACT

This paper presents ZigZag, an 802.11 receiver that combats hidden terminals. ZigZag exploits 802.11 retransmissions which, in the case of hidden terminals, cause successive collisions. Due to asynchrony, these collisions have different interference-free stretches at their start, which ZigZag uses to bootstrap its decoding.

ZigZag makes no changes to the 802.11 MAC and introduces no overhead when there are no collisions. But, when senders collide, ZigZag attains the same throughput as if the colliding packets were a priori scheduled in separate time slots. We build a prototype of ZigZag in GNU Radio. In a testbed of 14 USRP nodes, ZigZag reduces the average packet loss rate at hidden terminals from 82.3% to about 0.7%.

1 INTRODUCTION

Collisions and hidden terminals are known problem in 802.11 networks [9, 22, 19, 2, 23, 26, 37]. Measurements from a production WLAN show that 10% of the sender-receiver pairs experience severe packet loss due to collisions [9]. Current 802.11 WLANs rely on carrier sense (CSMA) to limit collisions–i.e., senders sense the medium and abstain from transmission when the medium is busy. This approach is successful in many scenarios, but when it fails, as in the case of hidden terminals, the impact on the interfering senders is drastic; the senders either repeatedly collide and their throughputs plummet, or one sender captures the medium preventing the other from getting packets through [22, 19, 37]. The 802.11 standard proposes the use of RTS-CTS to counter collisions, but experimental results show that enabling RTS-CTS significantly reduces the overall throughput [19, 37, 40, 2], and hence WLAN deployments and access point (AP) manufacturers disable RTS-CTS by default [1, 29]. Ideally, one would like to address this problem without changing the 802.11 MAC or affecting senders that do not suffer from hidden terminals.

This paper introduces ZigZag, a new 802.11 receiver that increases WLAN resilience to collisions. ZigZag requires no changes to the 802.11 MAC and introduces no overheard in the case of no collision. In fact, in the absence of collisions, ZigZag acts like a typical 802.11 receiver. But, when senders collide, ZigZag achieves the same performance as if the colliding packets were a priori scheduled in separate time slots.

ZigZag exploits a subtle opportunity for resolving collisions, an opportunity that arises from two basic characteristics of 802.11:

1. An 802.11 sender retransmits a packet until it is acked or timed out, and hence when two senders collide they tend to collide again on the same packets.

2. 802.11 senders jitter every transmission by a short random interval,[1] and hence collisions start with a random stretch of interference free bits.

To see how ZigZag works, consider the hidden terminal scenario in Fig. 1, where Alice and Bob, unable to sense each other, transmit simultaneously to the AP, causing collisions. When Alice's packet collides with Bob's, both senders retransmit their packets causing a second collision, as shown in Fig. 2. Further, because of 802.11 random jitters, the two collisions are likely to have different offsets, i.e., $\Delta_1 \neq \Delta_2$. Say that the AP can compute these offsets (as explained in §5.1), the AP can then find a chunk of bits that experience interference in one collision but is interference-free in the other, such as chunk 1 in Fig. 2. A ZigZag AP uses this chunk to bootstrap its decoder. In particular, since chunk 1 is interference-free in the first collision, the AP can decode it using a standard decoder. The AP then subtracts chunk 1 from the second collision to decode chunk 2. Now, it can go back to the first collision, subtract chunk 2, decode chunk 3, and proceed until both packets are fully decoded.

ZigZag is a novel approach to decoding collisions, different from prior work on interference cancellation [34, 17] and joint decoding [32]. Basic results on the capacity of the multi-user channel show that if the two hidden terminals transmit at the rate supported by the medium in the absence of interference, i.e., rate R in Fig. 3, the aggregate information rate in a collision, being as high as $2R$, exceeds capacity, precluding any decoding [32, 12]. Thus, interference cancellation and joint decoding, designed for cellular networks with non-bursty traffic and known users [34, 5], have a fundamental limitation when applied in 802.11 networks: they require a sender to change the way it modulates and codes a packet according to whether the packet will collide or not. This leaves 802.11 senders with the following tradeoff: either they tune to a suboptimal rate that works in the presence of collision, though not every packet will collide, or they send

[1] Each transmission picks a random slot between 0 and *CW* [38].

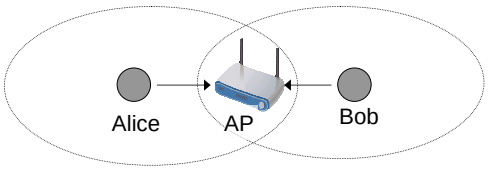


Figure 1—A Hidden Terminals Scenario.

at the best rate in the absence of collision, but accept that the network cannot use these methods to resolve collisions. In contrast, with ZigZag, the senders need not make such a tradeoff. ZigZag allows the senders to transmit at the best rate supported by the medium in the absence of collisions, R. However, if collisions occur, ZigZag decodes pairs of collisions that contain the same packets. The average information rate in such a collision pair is $2R/2 = R$. This rate is both decodable and as efficient as if the two packets were scheduled in separate time slots.

ZigZag has the following key features.

- *It works with various modulations:* When its a chunk's turn to be decoded, the chunk has already been rid of interference. Thus, ZigZag can employ a standard 802.11 decoder as a black-box to decode the chunks, which allows it to work with collisions independent of their underlying modulation scheme (i.e., bit rate).
- *It is backward compatible:* A ZigZag receiver can operate with unmodified 802.11 senders and requires no changes to the 802.11 protocol (see §7 for how to send acks).
- *It generalizes to more than a pair of colliding packets*, as explained in §8 and experimentally demonstrated in §10.6.
- *It has a lower bit error rate than if the packets were sent in separate time slots*. This might sound surprising, but this is possible because every bit is received twice, once in every collision, and thus has twice as much chance to be decoded correctly. ZigZag applies the decoding algorithm both in the forward and backward directions and combines the results to reduce decoding errors.

We have implemented a ZigZag prototype in GNU Radio, and evaluated it in a 14-node testbed, where 12% of the sender-receiver pairs are hidden terminals, 8% sense each other partially, and 80% sense each other perfectly. Our results reveal the following findings.

- The loss rate averaged over scenarios with partial or perfect hidden terminals decreases from 82.3% to less than 0.7%, with some severe cases where the loss rate goes down from 100% to zero.
- Averaging over all sender-receiver pairs, including those that do not suffer from hidden terminals, we find that ZigZag improves the average throughput by 31% when compared to current 802.11.
- At all SNRs, ZigZag's bit error rate (BER) is lower than if the colliding packets were scheduled in separate time slots. The average reduction in bit error in comparison to scheduling packets separately is 1.4x.

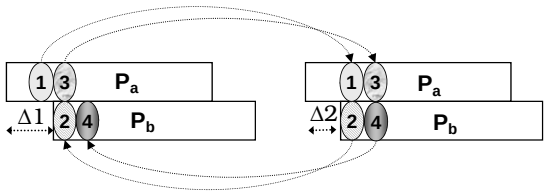


Figure 2—ZigZag Decoding. ZigZag decodes first chunk 1 in the first collision, which is interference free. It then subtracts chunk 1 from the second collision and decodes chunk 2, which it can then subtracts from the first collision and decodes chunk 3, etc.

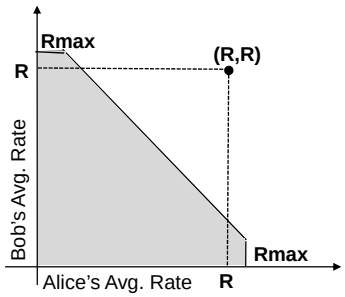


Figure 3—Interference Cancellation and Joint Decoding Require Inefficient Rates. The figure shows the classic illustration of the capacity region of the multi-user channel. Points outside the shaded area are undecodable because the combined rate of the two senders exceeds the capacity. If Alice and Bob transmit close to the best rate supported by the medium in the absence of interference, R, their combined rates will be (R, R), which is outside the capacity region, and hence cannot be decoded.

2 RELATED WORK

Related work falls in the following two areas.

(a) Collisions in WLAN and Mesh Networks. The closest to our work is by Halperin et al. [16] who articulate the benefits of decoding 802.11 collisions. ZigZag however is significantly different from the approach in [16]. Halperin et al. use joint decoding, which, as explained in §1, requires the senders to transmit a priori at the low rate required for decoding in the presence of collisions, though not every packet will collide. Additionally, the system works by modeling the collision signal. The complexity of such a model increases significantly at high modulation schemes, and is also exponential in the number of colliding packets. In contrast, ZigZag does not require the senders to send differently depending on whether a packet will collide, can work with various 802.11 modulations, and is linearly extendable to more than a pair of colliding packets.

Our work is also related to analog network coding (ANC) [21]. ANC, however, does not deal with general collisions or hidden terminals. An ANC receiver can decode collisions only if it already knows one of the two colliding packets. In principle, one can combine ANC and ZigZag to create a system that addresses hidden terminals, and collects network coding gains.

Additionally, prior works have studied wireless interference [30, 15, 9, 22, 19, 2, 23, 26, 37], and proposed MAC modifications to increase resilience to collisions [41, 11, 20, 6, 4, 28]. In comparison, this paper presents mechanisms that decode collisions rather than avoiding them, and works within the 802.11 MAC rather than proposing a new MAC.

(b) Communication and Information Theory: The idea of decoding interfering users has received much interest in information and communications theories [32, 34, 8, 33, 36, 35]. The main feature that distinguishes ZigZag from prior works in those areas is that ZigZag resolves 802.11 collisions without requiring any scheduling, power control, synchronization assumptions, or new codes.

Among the deployed systems, CDMA receivers decode a user by treating all other users as noise [8]. A CDMA solution for hidden terminals in WLANs, however, would require major changes to 802.11 including the use of power control and special codes [5, 8]. Furthermore, CDMA is known to be highly suboptimal in high SNR regimes (e.g., worse than TDMA [32]), which are typical in WLANs [14].

Finally, interference cancellation is a known approach for decoding interfering users in CDMA cellular networks [5]. Interference cancellation applies only under specific constraints. As stated in §1, the senders' information rates must stay below capacity. Additionally, practical systems require either that the interfering senders have significantly different powers [34], or they have different levels of coding [17, 32]. ZigZag includes interference cancellation as a special case, and uses it only when the senders' powers and rates permit. ZigZag, however, does not rely on interference cancellation as the main means of decoding and thus works when interference cancellation does not apply.

3 SCOPE

ZigZag is a new 802.11 receiver that can decode collisions. Its design is focused on addressing hidden terminals in WLANs. ZigZag's benefits extend to mesh networks, where having receivers that can decode collisions could enable more concurrent transmissions and hence higher spatial reuse. Exploring mesh benefits is, however, beyond the scope of this paper.

ZigZag adopts a best effort design; in the absence of collisions it acts like current 802.11 receivers, but when collisions occur it tries to decode them. Of course there are scenarios where collision decoding may fail, but since ZigZag does not introduce any overhead for the case of no collision, its presence can only increase the throughput of the WLAN. In §7, we explain how one can deploy ZigZag in a WLAN by changing only the access points and without modifying the clients.

ZigZag resolves a variety of collision patterns. The main idea underlying its decoding algorithm is to find a collision free chunk, which it exploits to bootstrap the decoding process. Once the decoder is bootstrapped the process is iterative and at each stage it produces a new interference-free chunk, decodable using standard decoders. For example, as explained in §1, ZigZag can decode the pattern in Fig. 2 by decoding first chunk 1 in the first collision, and subtracting it from the second collision, obtaining chunk 2, which it decodes and subtracts from the first collision, etc. Using the same principle, ZigZag can decode other patterns like those in Fig. 4. In particular, it can decode patterns where the collisions overlap as in Fig. 4a, and patterns in which colliding packets change order as in Fig. 4b, or even patterns where the packets have different sizes, as in Fig. 4c.

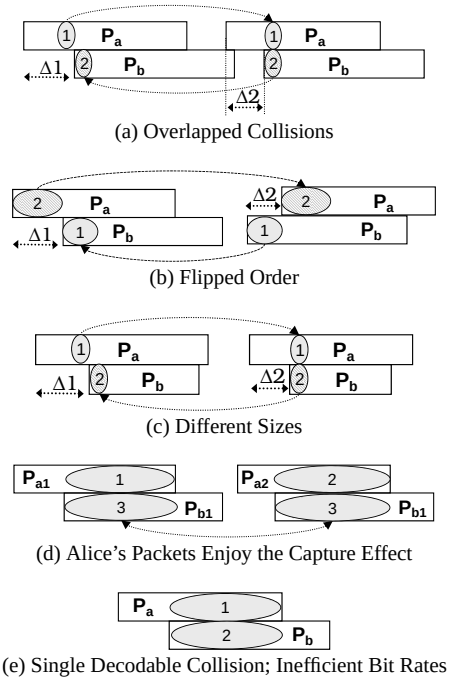


Figure 4—ZigZag applies to various collision patterns. The figure shows a variety of collision patterns that ZigZag resolves. The top three patterns are decoded chunk-by-chunk. The forth pattern refers to a capture effect which occurs because Alice's power at the AP is significantly higher than Bob's. The last pattern occurs when Alice's power is significantly higher than Bob's, but Bob's power is also significantly higher than necessary for his bit rate.

ZigZag exploits collision patterns that arise from capture effects. Say that Alice's power at the AP is significantly higher than Bob's, and hence her packets enjoy the capture effect [37]. Currently such a scenario translates into significant unfairness to Bob whose packets do not get through [22, 19, 37]. Like current APs, a ZigZag AP decodes every packet from Alice, the high power sender. Unlike current APs however, ZigZag subtracts Alice's packet from the collision signal and try to decode Bob's packet. However, if Alice's power is excessively high, even a small imperfection in subtracting her signal would contribute a significant noise to Bob's, preventing correct decoding of his packets. In this case, the next collision will involve a new packet from Alice and Bob's retransmission of the same packet, as shown in Fig. 4d. ZigZag decodes Alice's new packet and subtracts it to obtain a second version of Bob's packet, which may also contain errors. ZigZag however combines the two faulty versions of Bob's packet to correct the errors. This is done using Maximal Ratio Combining (MRC) [7], a classic method for combining information from two receptions to correct for bit errors.[2]

[2]To get a feel for how MRC works, consider the case where the senders use the BPSK modulation, which maps a "0" bit to -1 and a "1" bit to +1. If the AP receives two versions of the i^{th} bit. The first version is -0.2 and the second is +0.5, then assuming the channel has not changed between the two receptions, MRC estimates the bit as the average of these two receptions

Finally, whenever the powers permit, ZigZag decodes patterns that involve a single collision like those in Fig. 4e. This occurs when Alice's power is significantly higher than Bob's, and both senders happen to transmit at a bit rate lower than the best rate supported by the channel. In this case, ZigZag can apply interference cancellation [34], i.e., ZigZag decodes P_a and subtracts it from the received signal to decode P_b, decoding both packets using a single collision.

ZigZag can also decode collisions that involve more than a pair of packets, which we discuss in §8.

4 A COMMUNICATION PRIMER

A wireless signal is typically represented as a stream of discrete complex numbers [27]. To transmit a packet over the wireless channel, the transmitter maps the bits into complex symbols, in a process called modulation. For example, the BPSK modulation (used in 802.11 at low rates) maps a "0" bit to $e^{j\pi} = -1$ and a "1" bit to $e^{j0} = 1$. The transmitter generates a complex symbol every T seconds. In this paper, we use the term $\mathbf{x}[n]$ to denote the complex number that represents the n^{th} transmitted symbol.

The received signal is also represented as a stream of complex symbols spaced by the sampling interval T. These symbols differ, however, from the transmitted symbols, both in amplitude and phase. In particular, if the transmitted symbol is $\mathbf{x}[n]$ the received symbol can be approximated as:

$$\mathbf{y}[n] = \mathbf{H}\mathbf{x}[n] + \mathbf{w}[n], \quad (1)$$

where $\mathbf{H} = he^{\gamma}$ is also a complex number, whose magnitude h refers to channel attenuation and its angle γ is a phase shift that depends on the distance between the transmitter and the receiver, and $\mathbf{w}[n]$ is a random complex noise.[3]

If Alice and Bob transmit concurrently their signals add up, and the received signal can be expressed as:

$$\mathbf{y}[n] = \mathbf{y}_A[n] + \mathbf{y}_B[n] + \mathbf{w}[n],$$

where $\mathbf{y}_A[n] = \mathbf{H}_A\mathbf{x}[n]$ and $\mathbf{y}_B[n] = \mathbf{H}_B\mathbf{x}_B[n]$ refer to Alice's and Bob's signals after traversing their corresponding channels to the AP. Note that the above does not mean that we assume the n^{th} symbol from Alice combines with the n^{th} symbol from Bob. The notation is only to keep the exposition clear.

4.1 Practical Issues

A few practical issues complicates the process of estimating the transmitted symbols from the received symbols: frequency offset, sampling offset, and inter-symbol interference. Typically, a decoder has built-in mechanisms to deal with these issues [27].

(a) Frequency Offset and Phase Tracking: It is virtually impossible to manufacture two radios centered at the same exact frequency. Hence, there is always a small frequency difference, δf, between transmitter and receiver. The frequency offset causes a linear displacement in the phase of the received signal that increases over time, i.e.,

$$\mathbf{y}[n] = \mathbf{H}\mathbf{x}[n]e^{j2\pi n\delta fT} + \mathbf{w}[n].$$

Typically, the receiver estimates δf and compensates for it.

(b) Sampling Offset: The transmitted signal is a sequence of complex samples separated by a period T. However, when transmitted on the wireless medium, these discrete values have to be interpolated into a continuous signal. The continuous signal is equal to the original discrete samples, only if sampled at the exact same positions where the discrete values were. Due to lack of synchronization, a receiver cannot sample the received signal exactly at the right positions. There is always a sampling offset, μ. Further, the drift in the transmitter's and receiver's clocks results in a drift in the sampling offset. Hence, decoders have algorithms to estimate μ and track it over the duration of a packet.

(c) Inter-Symbol Interference (ISI) While Eq. 1 makes it look as if a received symbol $\mathbf{y}[n]$ depends only on the corresponding transmitted symbol $\mathbf{x}[n]$, in practice, neighboring symbols affect each other to some extent. Practical receivers apply linear equalizers [24] to mitigate the effect of ISI.

5 ZIGZAG DECODING

We explain ZigZag decoding using the hidden terminal scenario in Fig. 6, where Alice and Bob, not able to sense each other, transmit simultaneously to the AP, creating repeated collisions. Later in §8, we extend our approach to a larger number of colliding senders.

Like current 802.11, when a ZigZag receiver detects a packet it tries to decode it, assuming no collision, and using a typical decoder. If decoding fails (e.g., the decoder loses synchronization or the decoded packet does not satisfy the checksum), the ZigZag receiver will check whether the packet has suffered a collision, and proceed to apply ZigZag decoding.

5.1 Is It a Collision?

To detect a collision, the AP exploits that every 802.11 packet starts with a known preamble [38]. The AP detects a collision by correlating the known preamble with the received signal. Correlation is a popular technique in wireless receivers for detecting known signal patterns [8]. Say that the known preamble is L samples. The AP aligns these L samples with the first L received samples, computes the correlation, shifts the alignment by one sample and re-computes the correlation. The AP repeats this process until the end of the packet. The preamble is a pseudo-random sequence that is independent of shifted versions of itself, as well as Alice's and Bob's data. Hence the correlation is near zero except when the preamble is perfectly aligned with the beginning of a packet. Fig. 5 shows the correlation as a function of the position in the received signal. The measurements are collected

i.e., $(0.5 - 0.2)/2 = 0.1 > 0$, and hence it decodes the bit as a "1" bit. For further information about MRC and symbol combining methods, we refer the reader to [7, 39].

[3] This models flat-fading quasi-static channels.

using GNURadios (see §10). Note that when the correlation spikes in the middle of a reception, it indicates a collision. Further, the position of the spike corresponds to the beginning of the second packet, and hence shows Δ, the offset between the colliding packets.

The above argument is only partially correct because the frequency offset can destroy the correlation, unless the AP compensates for it. Assume that Alice's packet starts first and Bob's packet collides with it starting at position Δ. To detect Bob's colliding packet, the AP has to compensate for the frequency offset between Bob and itself. The frequency offset does not change over long periods, and thus the AP can maintain coarse estimates of the frequency offsets of active clients as obtained at the time of association. The AP uses these estimates in the computation.

Mathematically, the correlation is computed as follows. Let $\mathbf{y}$ be the received signal, which is the sum of the signal from Alice, $\mathbf{y}_A$, the signal from Bob, $\mathbf{y}_B$, and the noise term $\mathbf{w}$. Let the samples $\mathbf{s}[k], 1 \leq k \leq L$, refer to the known preamble, and $\mathbf{s}^*[k]$ be the complex conjugate. The correlation, Γ, at position Δ is:

$$\begin{aligned}\Gamma(\Delta) &= \sum_{k=1}^{L} \mathbf{s}^*[k]\mathbf{y}[k+\Delta] \\ &= \sum_{k=1}^{L} \mathbf{s}^*[k](\mathbf{y}_A[k+\Delta] + \mathbf{y}_B[k] + \mathbf{w}[k])\end{aligned}$$

The preamble, however, is independent of Alice's data and the noise, and thus the correlation between the preamble and these terms is about zero. Since Bob's first L samples are the same as the preamble, we obtain:

$$\begin{aligned}\Gamma(\Delta) &= \sum_{k=1}^{L} \mathbf{s}^*[k]\mathbf{y}_B[k] \\ &= \sum_{k=1}^{L} \mathbf{s}^*[k]\mathbf{H}_B\mathbf{s}[k]e^{j2\pi k\delta f_B T} \\ &= \mathbf{H}_B \sum_{k=1}^{L} |\mathbf{s}[k]|^2 e^{j2\pi k\delta f_B T}\end{aligned}$$

Since a frequency offset exists between Bob and the AP, i.e., $\delta f_B \neq 0$, the terms inside the sum have different angles and may cancel each other. Thus, the AP should compute the value of the correlation after compensating for the frequency offset, which we call Γ'. At position Δ this value becomes:

$$\begin{aligned}\Gamma'(\Delta) &= \mathbf{H}_B \sum_{k=1}^{L} |\mathbf{s}[k]|^2 e^{j2\pi k\delta f_B T} \times e^{-j2\pi k\delta f_B T} \\ &= \mathbf{H}_B \sum_{k=1}^{L} |\mathbf{s}[k]|^2.\end{aligned}$$

The magnitude of $\Gamma'(\Delta)$ is the sum of energy in the preamble, and thus it is significantly large, i.e., after compensating for the frequency offset, the magnitude of the correlation

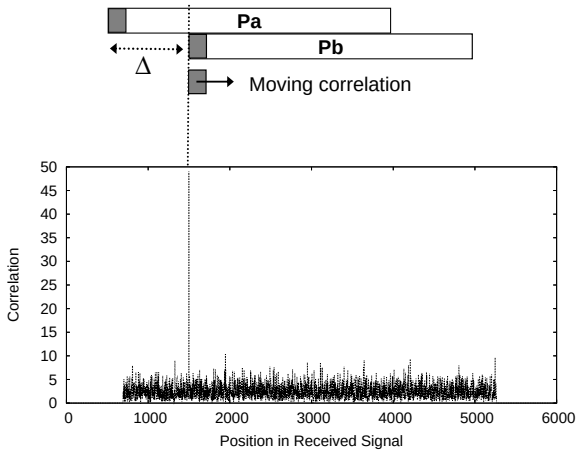


Figure 5—Detecting Collisions by Correlation with the Known Preamble. The correlation spikes when the correlated preamble sequence aligns with the preamble in Bob's packet. This allows the AP to detect the occurrence of a collision and where exactly it starts.

spikes when the preamble aligns with the beginning of Bob's packet, as shown in Fig. 5. Imposing a threshold enables us to detect whether the AP received a collision signal and where exactly the second packet starts.

5.2 Did the AP Receive Two Matching Collisions?

Now that it is clear that the received signal is the result of collision, the AP searches for a matching collision, i.e., a collision of the same two packets. The AP stores recent unmatched collisions (i.e., stores the received complex samples). It is sufficient to store the few most recent collisions because, in 802.11, colliding sources try to retransmit a failed transmission as soon as the medium is available [38].

We use the same correlation trick to match the current collision against prior collisions. Assume the AP is trying to match two collisions (P_1, P_2), and (P_1', P_2'). Without loss of generalization, let us focus on checking whether P_2 is the same as P_2'. The AP already knows the offset in each collision, i.e., Δ and Δ'. The AP aligns the two collisions at the positions where P_2 and P_2' start. If the two packets are the same, the samples aligned in such a way are highly dependent (they are the same except for noise and the retransmission flag in the 802.11 header), and thus the correlation spikes. If P_2 and P_2' are different, their data is not correlated and the correlation does not spike at that alignment.

5.3 How Does the AP Decode Matching Collisions?

Say that the AP found a pair of matching collisions like those in Fig. 6. Note that Fig. 6 is the same as Fig. 2 in the introduction except that we distinguish between two images of the same chunk that occur in different collisions, e.g., chunk 1 and chunk 1'. By now the AP knows the offsets Δ_1 and Δ_2, and hence it can identify all interference-free symbols and decode them using the standard method.

Next, the AP performs ZigZag decoding, which requires identifying a *bootstrapping chunk*, i.e., a sequence of symbols marred by interference in one collision and interference-free in the other. Say that the first collision has the larger off-

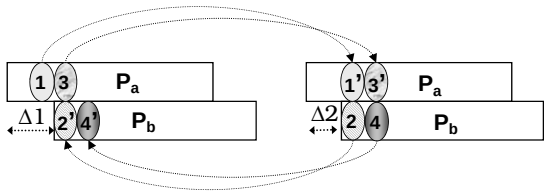


Figure 6—ZigZag decodes then re-encodes a chunk. Before subtracting a decoded chunk, like chunk 1, ZigZag needs to re-encode the bits to create an image of chunk 1', as received in the second collision.

set, i.e., $\Delta_1 > \Delta_2$, the bootstrapping chunk then is located in the first collision starting at position Δ_2 and has a length of $\Delta_1 - \Delta_2$ samples. This is chunk 1 in Fig. 6.

The rest of the decoding works iteratively chunk-by-chunk. In each iteration, the AP decodes a chunk, re-encodes the decoded symbols and subtract them from the other collision. For example, in Fig. 6, the AP decodes chunk 1 from the first collision, re-encodes the symbols in chunk 1 to create an image of chunk 1', which it subtracts from the second collision to obtain chunk 2. The AP iterates on the rest of the chunks as it did on chunk 1, until it is done decoding all chunks in the colliding packets.

(a) The Decoder. ZigZag can use any standard decoder as a black box. Specifically, the decoder operates on a chunk after it has been rid from interference, and hence can use standard techniques. This characteristic allows ZigZag to directly apply to any modulation scheme as it can use any standard decoder for that modulation as a black box. Further, the two colliding packets may use different modulation (different bit rates) without requiring any special treatment.

(b) Re-Encoding a Chunk. Now that the AP knows the symbols that Alice sent in chunk 1, it uses this knowledge to create an estimate of how these symbols would look after traversing Alice's channel to the AP, i.e., to create an image of chunk 1', which it can subtract from the second collision.

In §5.4 we explain how the AP computes channel parameters, but for now, let us assume that the AP knows Alice's channel, i.e., $\mathbf{H}_A$, δf_A, and μ_A. Denote the symbols in chunk 1 by $\mathbf{x}_A[n] \ldots \mathbf{x}_A[n+K]$. A symbol that Alice sends, $\mathbf{x}_A[n]$, is transformed by the channel to $\mathbf{y}_A[n]$ where:

$$\mathbf{y}_A[n] = \mathbf{H}_A \mathbf{x}_A[n] e^{j2\pi\delta f_A T}. \quad (2)$$

The AP would have received $\mathbf{y}_A[n]$ had it sampled the signal exactly at the same locations as Alice. Because of sampling offset, the AP samples the received signal μ_A seconds away from Alice's samples. Thus, given the samples $\mathbf{y}_A[n] \ldots \mathbf{y}_A[n+K]$, the AP has to interpolate to find the samples at $\mathbf{y}_A[n+\mu_A] \ldots \mathbf{y}_A[n+K+\mu_A]$.

To do so, we leverage the fact that we have a band-limited signal sampled according to the Nyquist criterion. Nyquist says that under these conditions, one can interpolate the signal at any discrete position, e.g., $n+\mu_A$, with complete accuracy, using the following equation [27]:

$$\mathbf{y}_A[n+\mu_A] = \sum_{i=-\infty}^{\infty} \mathbf{y}_A[i] sinc(\pi(n+\mu_A - i)),$$

where *sinc* is the sinc function. In practice, the above equation is approximated by taking the summation over few symbols (about 8 symbols) in the neighborhood of n.

Now that the AP has an image of chunk 1' as received, it subtracts it from the second collision to obtain chunk 2, and proceeds to repeat the same process on this latter chunk.

5.4 Estimating and Tracking the System Parameters

The receiver has to estimate the system's parameters for both Alice and Bob using the preamble. Without loss of generality, we focus our discussion on Bob, i.e., we focus on the sender that starts second. This is the harder case since the preamble in Bob's packet, typically used for channel estimation, is immersed in noise. We need to learn $\mathbf{H}_B$, μ_B, and δf_B.

(a) Channel. Again we play our correlation trick, i.e., we correlate the received samples with the known preamble. Recall that the correlation at the peak is:

$$\Gamma'(\Delta) = \mathbf{H}_B \sum_{k=1}^{L} |\mathbf{s}[k]|^2.$$

The AP knows the magnitude of the transmitted preamble i.e., it knows $|\mathbf{s}[k]|^2$. Hence, once it finds the maximum value of the correlation over the collision, it substitutes in the above equation to compute $\mathbf{H}_B$.

(b) Frequency Offset. The frequency offset does not change significantly over a long period. Since decoders already estimate the frequency offset, an initial coarse estimate can be computed using any prior interference free packet from the client (e.g., the association packet).

However, this coarse estimate is not sufficient since any residual errors in estimating δf translate into linear displacement in the phase that accumulates over the duration of a packet. Any typical decoder tracks the signal phase and corrects for the residual errors in the frequency offset. Since ZigZag uses a typical decoder as a black box, it need not worry about tracking the phase while decoding. Additionally, as it reconstructs an image of a received chunk, ZigZag tracks the phase in the reconstructed image of a chunk. Consider as an example, reconstructing an image of chunk 1'. First we reconstruct the image using the current estimate of the frequency offset, as explained in §5.3(b). Next we subtract that image from the second collisions to get chunk 2. Now, we reconstruct chunk 2 and subtracted from the second collision, creating an estimate of chunk 1', which we term chunk 1". We compare the phases in chunk 1' and chunk 1". The difference in the phase is caused by the residual error in our estimate of the frequency offset. We update our estimate of the frequency offset as follows:

$$\delta f = \delta f + \alpha \delta\phi / \delta t,$$

where α is just a small multiplier, $\delta\phi$ is the phase error which accumulated over a period δt.

(c) Sampling Offset. The procedure used to update and track the sampling offset is fairly similar to that used to update and track the frequency offset. Namely, the black-box

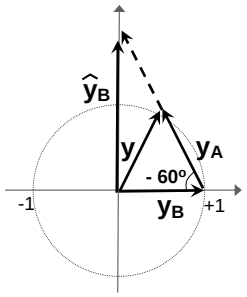


Figure 7—Errors Die Exponentially Fast. The error causes the AP to sum $\mathbf{y}_A$ instead of subtracting it. Hence, the error propagates from $\mathbf{y}_A$ to the estimate $\hat{\mathbf{y}}_B$, i.e., from one chunk to the next, only when the angle between the two vectors is smaller than 60^o, which occurs with probability $\frac{1}{3}$.

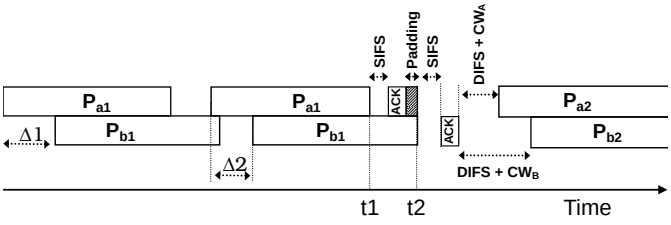


Figure 8—ACKing. The figure shows how ZigZag can send 802.11 synchronous acks.

decoder naturally tracks sampling offset when decoding a chunk. When reconstructing the image of a chunk, like chunk 1', we use the differences between chunk 1' and 1" to estimate the residual error in the sampling offset and track it.[4]

(d) Inter-Symbol Interference. When we reconstruct a chunk to subtract it from the received signal, we need to create as close an image of the received version of that chunk as possible. This includes any distortion that the chunk experienced because of multipath effects, hardware distortion, filters, etc. To do so, we need to invert the linear filter (i.e., the equalizer) that a typical decoder uses to remove these effects. The filter takes as input the decoded symbols before removing ISI, and produces their ISI-free version, as follows:

$$\mathbf{x}[i] = \sum_{l=-L}^{L} h_l \, \mathbf{x}_{ISI}[i+l],$$

where the h_l's are known as the filter taps. For our purpose, we can take the filter from the decoder and invert it. We apply the inverse filter to the symbols $\mathbf{x}[n]$ before using them in Eq. 2 to ensure that our reconstructed image of a chunk incorporates these distortions.

6 Dealing with Errors

Up to now, we have described the system assuming correct decoding. But what happens if the AP makes a mistake in decoding a symbol? For example, in Fig. 6, say the AP mistakenly decodes the first bit in chunk 1 as a "0" bit, when it is actually a "1" bit. Since chunk 1 is subtracted from the second collision to obtain chunk 2, the error will affect the first symbol in chunk 2. This in turn will affect the first symbol in chunk 3, and so on. Does that mean that a decoding error in one chunk propagates to subsequent chunks?

In the rest of this section, we will show the following:

- If a symbol error occurs while decoding, it may affect later chunks, but this propagation does not persist but rather decays exponentially fast.
- If the receiver applies ZigZag decoding on both the forward and backward directions and combines the results, the decoding error is less than if the two packets were sent in separate time slots.

(a) Errors Die Exponentially Fast. Assume the AP makes a mistake in decoding some symbol $\mathbf{y}_A$, and tries to use the erroneous symbol to decode $\mathbf{y}_B$ by subtracting the decoded vector from the received signal $\mathbf{y} = \mathbf{y}_A + \mathbf{y}_B$.[5] Say that the senders use the BPSK modulation and recall that BPSK maps a "0" bit to -1 and a "1" bit to +1. Let us see how such error affects BPSK.

In the worst case, and as shown in Fig. 7, the error causes the AP to add the vector instead of subtracting it, and hence will estimate $\hat{\mathbf{y}}_B$ as $\mathbf{y} + \mathbf{y}_A = \mathbf{y}_B + 2\mathbf{y}_A$. In BPSK, the AP will decode $\mathbf{y}_B$ to the wrong bit only if the estimate $\hat{\mathbf{y}}_B$ has the opposite sign as the original vector. This will happen only if the angle between the two vectors $\mathbf{y}_B$ and $\mathbf{y}_A$ is less than -60^o. Since the vectors $\mathbf{y}_B$ and $\mathbf{y}_A$ are independent, they can have any angle with respect to each other. Thus, the error occurs with probability less than $\frac{60}{180} = \frac{1}{3}$. Thus, in BPSK, errors die exponentially fast at a rate $\frac{2}{3}$.

Similarly, we can show exponential error decay for other modulations (4-QAM, 16-QAM, etc.).

(b) Forward and Backward Decoding. The ZigZag algorithm described so far decodes forward. In Fig. 2, it starts with chunk 1 in the first collision and proceeds until both packets are decoded. However, clearly the figure is symmetric. The AP could wait until it received all samples, and start decoding backward. If the AP does so, it will have two estimates for each symbol. It combines these estimates to reduce errors using MRC [7, 39], a classic method for diversity combining. In practice, we do not decode all the way forward and then all the way backward. We do it on a chunk-by-chunk basis, using the most recently decoded chunk as a bootstrapping chunk for backward decoding.

Our experimental results in §10.3 show that the combination of forward and backward decoding produces less errors than if the two colliding packets were sent in their separate slots. This may sound surprising at first. However, since every symbol gets sent twice (in the first and the second collisions), it has a better chance to be decoded correctly. The forward and backward decoding, exploits this by obtaining two copies of every symbol, one in the forward pass and the other in the backward. Combining these two copies, allows us to be more resilient to decoding errors than if the two packets are sent in separate time slots.

[4] The error in the sampling offset is computed using the Muller-and-Muller algorithm [27].

[5] We ignore the noise term $\mathbf{w}$ since it has a random effect on the error and can equally emphasize it or correct it.

7 BACKWARD COMPATIBILITY

It would be beneficial if ZigZag decoding requires no changes to senders. In this case, one can improve resilience to interference in a WLAN by purely changing the APs, and without requiring any modifications to the clients (e.g., laptops, PCs, PDAs). Compatibility with unmodified 802.11 senders requires a ZigZag receiver to ack the colliding senders once it decoded their packets; otherwise the senders will retransmit again unnecessarily. Recall that an 802.11 sender expects the ack to follow the packet, separated only by a short interval called SIFS [38]; Can a ZigZag receiver satisfy such requirement?

The short answer is "yes, with a high probability." To see how, consider again the example where Alice and Bob are hidden terminals, and say that the AP uses ZigZag to decode two of their packets, P_{a1} and P_{b1}, as shown in Fig. 8. The AP acks the packets according to the scheme outlined in Fig. 8. Specifically, by time t_1, the AP has fully decoded both P_{a1} and P_{b1}. Even more, by t_1 the AP has performed both forward-decoding and backward decoding for all bits transmitted so far, i.e., all bits except the few bits at the end of P_{b1}.[6] Thus, at t_1 the AP declares both packets decoded. It waits for a SIFS and acks packet P_{a1}. Though the ack collides with the tail of packet P_{b1}, the ack will be received correctly because Alice cannot hear Bob's transmission. Bob too will not be disturbed by the AP's ack to Alice because practical transmitters cannot receive and transmit at the same time. The AP then transmits some random signal to prevent Alice from transmitting her next packet, P_{a2}, before Bob's packet is acked. The AP knows how long this padding signal should be since it already has a decoded version of Bob's packet and knows its length. After Bob finishes his transmission the AP acks him as well.

One question remains, however, would the offset between the two colliding packets suffice to send an ack? Said differently, in Fig. 8, how likely is it that $t_2 - t_1 > SIFS + ACK$. If this is unlikely, the AP cannot send both acks synchronously. One can show that, given 802.11 standard timing, the likelihood that the time offset between the two packets is sufficient to send an ack is quite high. We can easily compute this likelihood for the different versions of 802.11. For the common deployment of backward compatible 802.11g, we prove in the appendix the following.

LEMMA 7.1. *In 802.11g, the probability that the time offset between two colliding packets is sufficient for sending an ACK is higher than 93.7%.*

Thus, a ZigZag receiver can resolve most collisions without any modification on the sender side. If the senders can be modified, ZigZag uses this to reduce the above probability to zero. Specifically, a ZigZag AP identifies ZigZag-aware senders during association. The AP always tries to send synchronous acks but if that fails and the sender is ZigZag-aware, the AP sends the ack asynchronously.

[6] This assumes that the receiver tries in parallel to use standard decoding and ZigZag, and takes whichever succeed and passing the checksum.

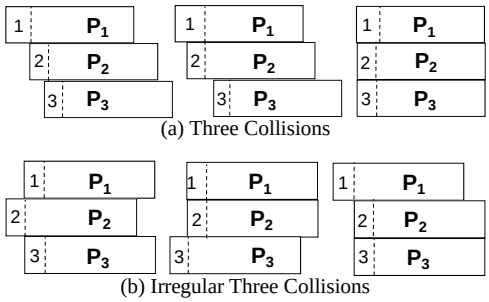


Figure 9—Applying ZigZag to Three Collisions.

8 BEYOND TWO INTERFERERS

Our description, so far, has been limited to a pair of colliding packets. ZigZag, however, can resolve a larger number of colliding senders. We start by showing via an example how to extend ZigZag to deal with three colliding senders. We then generalize the approach to many senders.

Consider the scenario in Fig. 9a, where we have three collisions from three different senders. We refer to the colliding packets by P_1, P_2 and P_3, and collision signals by C_1, C_2 and C_3. The figure shows a possible decoding order. We can start by decoding chunk 1 in the first collision, C_1, and subtract it from C_2 and C_3. As a result, chunk 2 in C_2 becomes interference-free and thus decodable. Next, we subtract chunk 2 from both C_1 and C_3. Now, chunk 3 in C_3 becomes interference-free; so we decode it and subtract it from both C_1 and C_2. One can use a similar approach to the three collisions in Fig. 9b. The idea is to find a decoding order such that, at each point, at least one of the three collisions has an interference-free chunk ready for decoding.

But, can we always find a decoding order that works? It turns out that, as long as the collisions satisfies the following conditions, there will be a chunk decoding order that works.

- There is a chunk in one of the collisions that is interference free.
- If P_i denotes the packet from the i^{th} transmitter, then for any k-subset of the packets $\{P_1, \cdots, P_n\}$, there exists k collisions, $\{C_1, C_2, \cdots, C_k\}$ such that the packets have combined differently (in terms of offsets) in these k collisions.

This is analogous to a linear system of n equations and n unknowns. The collisions are the linear equations, whereas the packets are the n unknowns. The system is solvable if the equations are linearly independent, i.e., one cannot derive one collision by linearly combining the other collisions.

The greedy algorithm below finds a chunk decoding order for any number of collisions that satisfy the above conditions.

- **Step 1:** For each of the collisions, decode all the overhanging chunks that are interference-free.
- **Step 2:** Subtract the known chunks wherever they appear in all collisions.

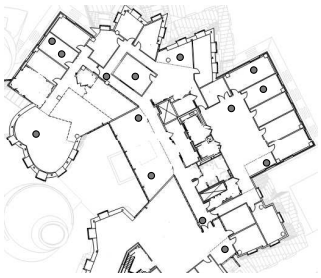


Figure 10—Testbed Topology. The dots refer to GNURadio nodes.

- **Step 3:** Decode all the new chunks that become interference free as a result of Step 2.
- Repeat the last two steps until all the chunks from all the packets are decoded.

We can prove the following lemma.

LEMMA 8.1. *As long as the above conditions are satisfied and there are undecoded chunks left out,* **Step 3** *always find a new interference free chunk.*

We omit the proof of the above lemma for space limitations. In practice, imperfections in the implementation limit the maximum number of colliding senders that can be correctly decoded. In 10.6, we show experimental results for scenarios with three interfering senders.

9 COMPLEXITY

ZigZag is linear in the number of colliding senders. In comparison to current decoders, ZigZag requires only two parallel decoding lines so that it can decode two chunks in the same time that it would take a current decoder to decode one chunk. Furthermore, most of the components that ZigZag uses are typical to wireless receivers. ZigZag uses the decoders and the encoders as black-boxes. Correlation, tracking, and channel estimation are all typical functionalities in a wireless receiver [27, 8].

10 EXPERIMENTAL ENVIRONMENT

We evaluate ZigZag in a 14-node GNURadio testbed. The topology is shown in Fig. 10. Each node is a commodity PC connected to a USRP GNU radio [18]. Software radios implement all of the wireless communication system in software (modulation, coding, etc.), thus providing a suitable platform for evaluating new receiver designs.

(a) Hardware and Software Environment. We use the Universal Software Radio Peripheral (USRP) [18] for our RF frontend. USRP is a generic RF frontend developed specifically for the GNU Radio SDR. We use the RFX2400 daughterboards which operate in the 2.4GHz range. The software for the signal processing blocks is from the open source GNURadio project [10].

(b) Modulation. ZigZag uses a modulation/demodulation module as a black-box and hence can work with a variety of modulation schemes. Our implementation, however, uses Binary Phase Shift Keying, *BPSK*, which is the modulation scheme that 802.11 uses at low rates.

(c) Configuration Parameters. We use the default GNURadio configuration, i.e., on the transmitter side DAC Rate is 128*e*6 samples/s, Interpolation Rate is 128, number of samples per symbol is 2. On the receiver side, the ADC rate is 64*e*6 samples/s and the Decimation Rate is 64. Given the above parameters and a BPSK modulation, the resulting bit rate is 500kb/s. Each packet consists of a 32-bit preamble, a 1500-byte payload, and 32-bit CRC.

(d) Implementation Flow Control. On the sending side, the network interface pushes the packets to the GNU software blocks with no modifications. All the action is at the receiver. First, the packet is detected using standard methods built in the GNURadio software package. Second, we try to decode the packet using the standard approach (i.e., using the BPSK decoder in the GNURadio software). If standard decoding fails, we use the algorithm in §5.1 to detect whether the packet has experienced a collision, and where exactly the colliding packet starts. If a collision is detected, the receiver matches the packet against any recent reception, as explained in §5.2. If no match is found, the packet is stored in case it helps decoding a future collision. If a match is found, the receiver performs chunk-by-chunk decoding on the two collisions, as explained in §5.3. Note that even when the standard decoding succeeds we still check whether we can decode a second packet with lower power (i.e., a capture scenario).

(e) Compared Schemes. We compare the following:

- **ZigZag:** This is a ZigZag receiver as described in §5 augmented with the backward-decoding described in §6.
- **Current 802.11:** This approach uses the same underlying decoder as ZigZag but operates over individual packet.
- **Collision-Free Scheduler:** This approach also uses the same basic decoder but prevents interference altogether by scheduling each sender in a different time slot.

(f) Metrics. We employ the following metrics:

- *Bit Error Rate (BER):* The percentage of incorrect bits averaged over every 100 packets.
- *Packet Loss Rate (PER):* This is the percentage of incorrectly received packets. We consider a packet to be correctly received if the BER in that packet is less than 10^{-3}. This is in accordance with typical wireless design, which targets a maximum BER of 10^{-3} before coding (and 10^{-5} after coding) [3, 31].[7]
- *Throughput:* This is the number of delivered packets normalized by the GNU Radio transmission rate. Again a packet is considered delivered if the uncoded BER is less than 10^{-3}. In comparison to packet loss rate, the throughput is more resilient to hidden terminals in scenarios that exhibit capture effects. This is because the terminal that captures the medium transmits at full rate and gets its

[7]For example, 802.11a target packet error rate (PER) is 0.1 for a packet size of 8000 bits. Given a maximum uncoded BER of 10^{-3}, practical channel codes like BCH Code(127,99) and BCH Code(15,5) achieve the desired PER.

packets through, causing unfairness to the other sender, but little impact on the overall throughput.

10.1 Setup

Since ZigZag acts exactly like current 802.11 receivers except when a collision occurs, our evaluation focuses on scenarios with hidden terminals, except in §10.5 where we experiment with various nodes in the testbed irrespective of whether they are hidden terminals. In every run, two or more senders transmit 500 packets to an access point. The AP (i.e., the receiver) logs the received signal and the logs are processed offline with the evaluated receiver designs.

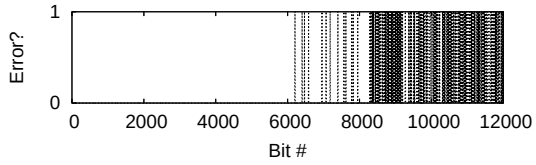
Software radios are incapable of accurately timing their carrier sense activity (CSMA) because they perform all signal processing functionalities in user mode on the PC. To approximate CSMA, we take the following measures. First, we setup an 802.11a node next to each of our USRP nodes. The objective is to create an 802.11a testbed that matches the topology in our USRP testbed but uses standard 802.11a cards, and copy the results of carrier sense from it to our USRP testbed.

For each USRP experiment, we check whether the corresponding 802.11a nodes can carrier sense each other. Specifically, we make each pair of the 802.11 nodes transmit at full speed to a third node considered as an AP, log the packets, and measure the percentage of packets each of them delivers to the AP. Next, we try to recreate the same behavior using the corresponding USRP nodes, where each packet that was delivered in the 802.11 experiments results in a packet delivery in the USRP experiments between the corresponding sender-receiver USRP pairs. Lost 802.11 packets are divided into two categories: collisions and errors. Specifically, a lost 802.11 packet that we can match with a loss from the concurrent sender is considered as a collision loss. Other losses are considered as medium errors and ignored. We try to make each USRP experiment match the collisions that occurred in the corresponding 802.11a experiment by triggering as many collisions as observed in the 802.11a traces. The USRP experiments are run without CSMA. Each run matches an 802.11 run between the corresponding 802.11 nodes. Each sender first transmits the same number of packets that the corresponding 802.11 correctly delivered in the matching 802.11 run. Then both senders transmit together as many packets as there were collision packets in the matching 802.11 run.

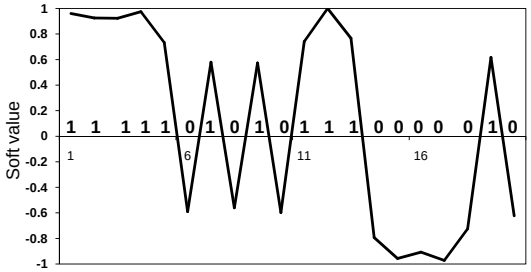
Software radios also cannot time 802.11 synchronous acks. Given the 802.11a traces, we know when a collision will occur, and that the sender should retry the packet, in which case the sender transmits each packet twice. However, if the ZigZag AP manages to decode using a single collision, we ignore the retransmission and do not count it against the throughput.

10.2 Micro-Evaluation

We examine the role of various components of ZigZag.



(a) Error Distribution due to Residual δf.



(b) ISI Prone Symbols

Figure 11—Effects of Residual Frequency Offset and ISI.

Table 1—Micro-Evaluation of ZigZag's components

Correlation	False Positives	3.1%	
	False Negatives	1.9%	
Frequency & Phase Tracking	Pkt size(Bytes)	800	1500
	Success With	99.6%	98.2%
	Success Without	89%	0%
ISI Filter	SNR	10dB	20dB
	Success With	99.6%	100%
	Success Without	47%	96%

(a) Correlation as a Collision Detector: We estimate the effectiveness of the correlation-based algorithm (§5.1) in detecting the occurrence of collisions. Our implementation sets the threshold to $\Gamma'(Delta) > \beta \times L \times SNR$, where β is a constant, L is the length of the preamble and SNR is a coarse estimate of the SNR of the colliding sender, which could be obtained from any previously decoded packets or from one of the sender's interference free chunks. For our testbed, β = 0.6-0.7 balances false positives with false negatives. Higher values eliminate false positives but make ZigZag miss some collisions, whereas lower values trigger collision-detection on clean packets. Note that neither false positives nor false negatives produce end-to-end errors. The harm of false positive is limited to computational resources, because in ZigZag marking a packet as a collision does not prevent correct decoding of that packet. The algorithm behaves as if the packet suffered capture effect and hence is decodable despite being marred by collision. False negatives, on the other hand, make ZigZag miss opportunities for decoding collisions but do not produce incorrect decoding. Our evaluation sets $\beta = 0.65$.

For SNRs in [6-20]dB, we run the collision detector on sets of 500 non-collision packets and 500 collisions, and report the results in Table 1. The average false positive rate (packets mistaken as collisions) is 3.1% and the average false negative rate (missing collisions) is 1.9%. Thus, the collision detector is pretty accurate for our purpose.

(b) Frequency and Phase Tracking: We evaluate the need for the frequency and phase tracking described in §5.4b. We disable our tracking algorithm (but leave the decoder unchanged) and provide the encoder with an initially accurate estimate of the frequency offset (as estimated by the decoder). We run ZigZag with and without tracking on 500 collision-pairs of 1500B packets. We find that without tracking none of the colliding packets is decodable (BER $> 10^{-3}$), whereas with tracking enabled, 98.2% of the colliding packets are decodable.

Fig. 11(a) explains this behavior. It plots the error as a function of the bit index in one of the colliding packets (black shades refer to errors). It shows that the first 6000 bits are decoded correctly, but as we go further the bits start getting flipped, and eventually most of the bits are in error. This is expected since even a small residual error in the frequency offset causes a phase rotation that increases linearly with time. Hence after some time the phase becomes completely wrong causing high decoding error rates. This effect is particularly bad for long packets since the errors accumulate over time. Table 1 shows that while ZigZag can decode 89% of the 800Byte packets without phase tracking, none of the 1500Byte packets is successfully decoded unless we enable phase tracking.

(c) Effect of ISI: Fig. 11(b), shows a snapshot of the ISI-affected received bits in our testbed. Recall that BPSK represents a "0" bit with -1 and a "1" bit with +1. The figure shows that the value of a received bit depends on the value of its neighboring bits. For example, a "1" bit tend to take a higher positive value if it is preceded by another "1", then if the preceding bit is a "0" bit.

We evaluate the importance of compensating for these distortions using the inverse filter described in §5.4d. We try to decode 500 collision pairs at different SNRs, with the filter on and off. Table 1 shows that while the filter is not important at relatively high SNRs, i.e., 20dB, it is necessary in low SNR regimes. This is expected as at low SNRs, the decoder has to combat both higher noise and ISI distortions.

10.3 Does ZigZag Decoding Work?

We start our evaluation with the basic hidden terminal scenario in Fig. 1, where Alice and Bob cannot sense each other and hence transmit simultaneously to the AP. We would like to check whether ZigZag decoding could make it look as if there were no interference and the two senders have been a priori allocated separate time slots. To do so, we look at the bit error rate (BER) as a function of SNR. This is a typical metric in designing wireless receivers [31, 3, 32]. Metrics like throughput or packet loss rate cannot distinguish between a completely erroneous packet and a packet that was discarded because of a single incorrect bit that could have been cheaply corrected with coding, whereas the BER provides a more detailed picture of the received data.

Fig. 12 plots the BER as a function of SNR. The scenario in our experiment is symmetric, i.e., Alice and Bob have the same SNR. The plots are only for ZigZag and the Collision-Free Scheduler because 802.11 in this scenario performed extremely poorly with BER close to 0.5. The figure reveals two basic results.

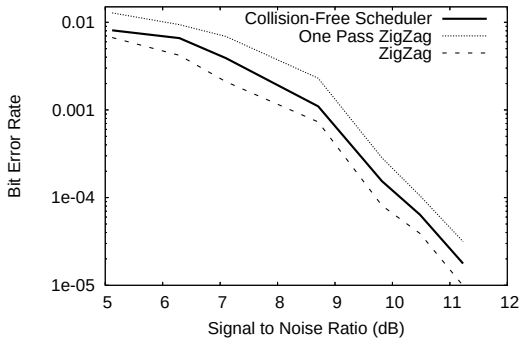


Figure 12—Comparison of Bit Error Rate (BER). ZigZag delivers packets that are as correct as if they were sent in separate time slots. Further, if both backward and forward decoding are used, the BER is lower than if collisions did not occur.

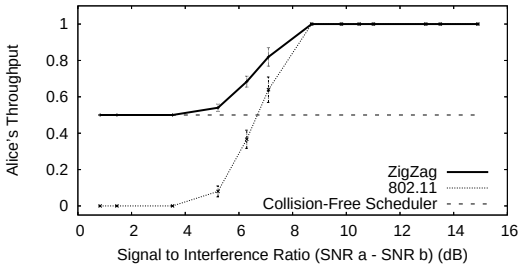
- At all SNRs, ZigZag decoding allows the receiver to decode collisions keeping the bit error close to the BER when the two packets are sent in separate time slots.
- With forward and backward decoding, the BER averaged over the explored SNRs is 1.4x lower than if we had no interference at all. Thus, two collisions are more resilient to bit errors than two packets sent in separate time slots.

10.4 Scenarios with Capture Effect

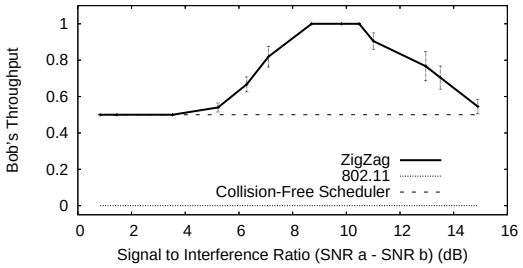
In contrast to the previous experiment where Alice and Bob have the same SNR at the AP, we now consider scenarios where one of the senders has a higher SNR, and thus can fully or partially capture the medium [25, 19]. Again we consider a scenario where Alice and Bob concurrently transmit to the same AP. We start from a setting where both senders are equal distance from the AP, i.e. $SNR_A = SNR_B$ and hence, $SINR = SNR_A - SNR_B = 0$. Gradually, we move Alice closer to the AP. As Alice moves closer, her SNR at the AP increases with respect to Bob's, making it easier for the AP to capture Alice's signal. We plot the result of this experiment in Fig. 13, for the case when the nodes use a Collision-Free Scheduler, current 802.11, and ZigZag.

Fig. 13 shows that ZigZag improves both throughput and fairness. In 802.11, when Alice and Bob are equal distance from the AP, their signals collide, and neither can be received. As Alice moves closer, her signal improves with respect to Bob's. When Alice's signal is 4-6 dB higher than Bob's, the capture effect starts, and we see a slight increase in Alice's throughput. As Alice gets even closer, Bob's signal becomes irrelevant. Note, however, that at all times Bob is never received at the AP with 802.11. In contrast, with the Collision-Free Scheduler, both Alice and Bob get a fair chance at accessing the AP. But the scheduler cannot exploit that as Alice gets closer, the capacity increases [32], making it possible to decode both Alice and Bob.

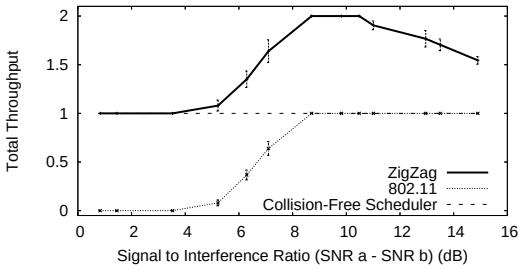
ZigZag outperforms both current 802.11 and the Collision-Free Scheduler. When Alice and Bob are equal dis-



(a) Alice's Throughput



(b) Bob's Throughput



(c) Total Throughput

Figure 13—Normalized Throughput in Scenarios with Capture Effects. The figure plots the throughput of the hidden terminals Alice and Bob, as Alice moves closer to the AP, i.e., as $SINR \approx SNR_A - SNR_B$ increases. It shows that ZigZag achieves higher throughput than both 802.11 and the Collision-Free Scheduler. ZigZag is also fairer than current 802.11 where Bob cannot get any packets through.

tance from the AP, it ensures that they are both received, as if they were allocated different time slots. As Alice moves closer to the AP, the capture effect starts kicking off. As a result, the AP can decode Alice's signal without the need for a second collision. The AP then subtracts Alice's signal from the collision and decode Bob's packet, and thus the total throughput becomes twice as much as the radio transmission rate. As Alice gets even closer, her signal completely covers Bob's signal making it impossible to decode Bob's packet. This experiment reveals the following:

- In scenarios with capture effects, ZigZag outperforms both 802.11 and the Collision-Free Scheduler.
- Neither 802.11 nor the Collision-Free Scheduler can benefit from scenarios where the network capacity is higher than the sum of the rates of the two senders. In contrast, ZigZag can exploit such scenarios to double the throughput of the network, decoding both hidden terminals using a single collision. Furthermore, ZigZag does not need to be explicitly informed of the capacity of the network to exploit it. It naturally transitions to exploit the increased capacity as the SNR increases.

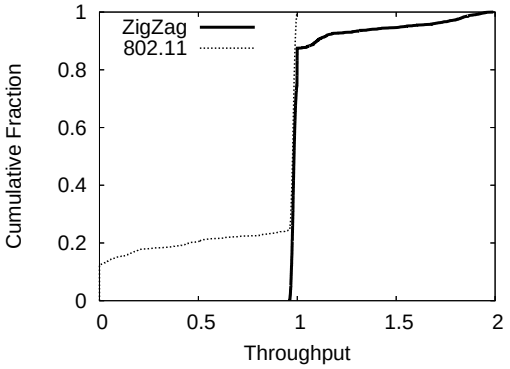


Figure 14—Normalized Throughput for the Whole Testbed. The figure shows a CDF of the throughputs in our testbed for pairs of competing flows, for both hidden and non-hidden terminal scenarios. ZigZag improves the average throughout in our testbed by 31%.

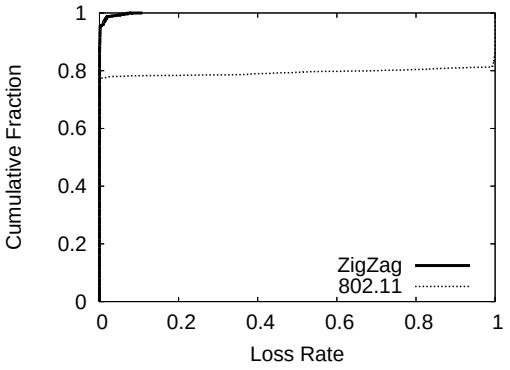


Figure 15—Loss Rate for the Whole Testbed. The figure shows a CDF of the packet loss rate in our testbed for pairs of competing flows, for both hidden and non-hidden terminal scenarios. ZigZag improves the average loss rate in our testbed from 18.9% to 0.2%.

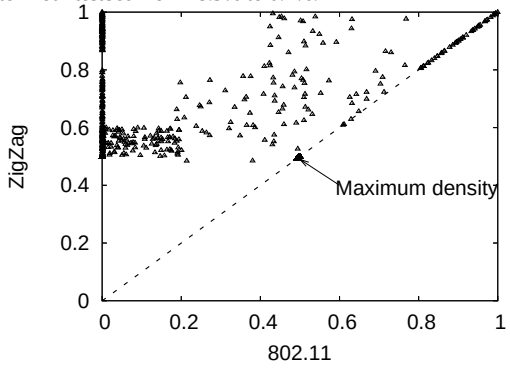


Figure 16—Scatter Plot of Flow Throughputs. The figure shows a scatter plot of ZigZag and 802.11 throughputs for each sampled sender-receiver pairs. ZigZag helps when there is a hidden terminal scenario but never hurts.

10.5 Testbed Throughput and Loss Rate

In this section, we measure how much ZigZag improves the performance in our indoor GNURadio testbed, shown in Fig. 10. The testbed has 14 nodes that form a variety of line-of-sight and none-line-of-sight topologies. While up to now we have focused only on scenarios with hidden terminals, in this section, we experiment with various testbed nodes irrespective of whether they are hidden terminals. Specifically, we pick two senders randomly. We pick an AP randomly from the nodes reachable by both senders. We mimic CSMA as explained in §10.1 and make each sender transmit

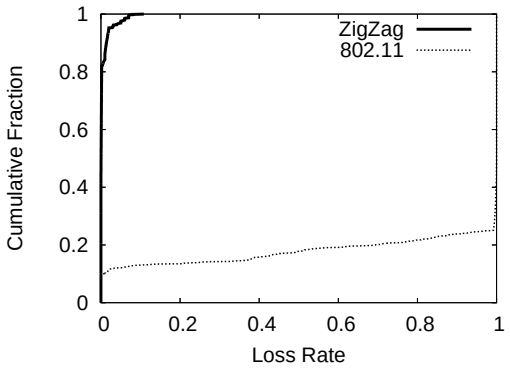


Figure 17—CDF of Loss Rate at Hidden Terminals. The figure zooms on scenarios with full or partial hidden terminals. ZigZag reduces the average loss rate for hidden terminals in our testbed from 82.3% to about 0.7%.

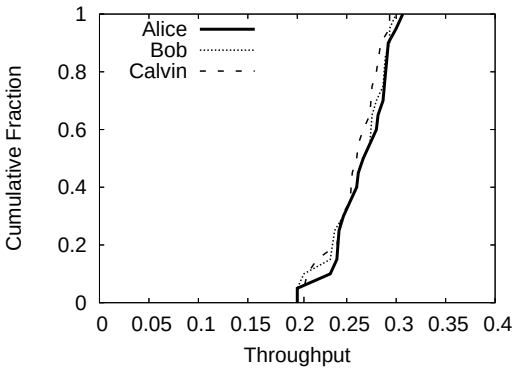


Figure 18—ZigZag's Performance with Three Hidden Terminals. Cumulative distribution of the throughput of three hidden terminals.

100 packets to the AP. We repeat the experiment with random set of sender pairs and different choice of APs. Among the sender pairs that we sampled 12% are perfect hidden terminals, 8% can sense each other partially, and 80% can sense each other perfectly.

First, we compare the throughput and loss rate under current 802.11 and ZigZag, for the whole network. Fig. 14 plots a CDF of the aggregate throughput, i.e., the sum of the throughput of each pair of concurrent senders. The figure shows that in our testbed, ZigZag increases the average throughput by 31%. This improvement arises from two factors. For all cases where the normalized aggregate throughput is less than 1, the improvement comes purely from ZigZag' s ability to resolve successive collisions. For cases where the aggregate throughput is higher than 1, the improvement is caused by a combination of being able to resolve a single collision whenever possible, and successive collisions otherwise. Note that interference cancellation applies only to cases whose throughputs are between 1.5 and 2, which are very few. Fig. 15 plots a CDF of the loss rates of individual sender-receiver pairs, i.e., the flows we experimented with. The figure shows that in our testbed, ZigZag reduces the average packet loss rate from 18.9% to 0.2%.

Next, we check that a ZigZag AP is always a conservative choice and does not hurt any flow. Fig. 16 shows a scatter plot of the throughout of every sender-receiver pair in our experiments, both under 802.11 and ZigZag. The figure shows that ZigZag consistently improves the throughput and does not hurt any sender-receiver pair.

Next, we zoom on the hidden terminals in our testbed, which we define as sender pairs that fail to sense each other fully or partially. Fig. 17 shows a CDF of the packet loss rate in transfers that suffered such hidden terminal scenarios. The figure shows that ZigZag improves the average loss rate for hidden terminals in our testbed from 82.3% to 0.7%. Furthermore, for some severe cases, the packet loss rate goes down from 99-100% to about zero.

10.6 Many Hidden Terminals

In §8 we generalized ZigZag to deal with more than a pair of colliding sources. Here, we evaluate how ZigZag performs on three collisions. In this experiment, we have three hidden terminals that transmit concurrently to a random AP. Fig. 18 shows the CDF of the throughput under ZigZag. The figure shows that all three senders see a fair throughput that is around one third of the medium throughput. Thus, even with more than a pair of colliding senders, ZigZag performs almost as if each of the colliding senders transmitted in a separate time slot.

11 Conclusion

The paper presents ZigZag, a new receiver design that decodes 802.11 collisions. It shows that ZigZag addresses the hidden terminal problem in WLANs, significantly improving the throughput and loss rate. We believe ZigZag can provide benefits in scenarios other than those explored in the paper. For example, it motivates a more aggressive MAC design that exploits concurrent transmissions in order to increase spatial reuse and network throughput. Also, recent work on analog network coding allows a receiver to decode collisions when it already knows one of the colliding packets. It seems plausible that ZigZag can be combined with analog network coding to improve concurrency, address hidden terminals, and collect network coding gains.

12 Acknowledgments

We thank Mythili Vutukuru, Chris Ng, Hari Balakrishnan, Szymon Chachulski and Ashish Khisti for their insightful comments.

References

[1] Broadcom Wireless LAN Adapter User Guide.

[2] Experimental Study of Hidden-node Problem in IEEE 802.11 Wireless Networks.

[3] ISL3873: Wireless LAN Integrated Medium Access Controller with Baseband Processor, 2000.

[4] N. Ahmed, V. Shrivastava, A. Mishra, S. Banerjee, S. Keshav, and K. Papagiannaki. Interference Mitigation in Wireless LANs using Speculative Scheduling (extended abstract). In *ACM Mobicom*, 2007.

[5] J. Andrews. Interference cancellation for cellular systems: A contemporary overview. *IEEE Wireless Communications*, 2005.

[6] V. Bharghavan, A. J. Demers, S. Shenker, and L. Zhang. MACAW: A Media Access Protocol for Wireless LAN's. In *ACM SIGCOMM 1994*.
[7] D. G. Brennan. On the Maximal Signal-to-Noise Ratio Realizable from Several Noisy Signals. *Proc. IRE*, 43:1530, October 1955.
[8] P. Castoldi. *Multiuser Detection in CDMA Mobile Terminals*. Artech house Publishers, 2002.
[9] Y.-C. Cheng, J. Bellardo, P. Benk, A. C. Snoeren, G. M. Voelker, and S. Savage. Jigsaw: solving the puzzle of enterprise 802.11 analysis. In *SIGCOMM*, 2006.
[10] G. FSF. Gnu radio - gnu fsf project. http://www.gnu.org/software/gnuradio.
[11] C. L. Fullmer and J. J. Garcia-Luna-Aceves. Solutions to Hidden Terminal Problems in Wireless Networks. In *SIGCOMM*, pages 39–49, 1997.
[12] R. G. Gallager. A Perspective on Multiaccess Channels. *IEEE Transactions on Information Theory*, IT-31(2), 1985.
[13] M. Gast. *802.11 Wireless Networks*. O'Reilly, 2005.
[14] J. Geier. Snr cutoff recommendations, 2005. http://www.wi-fiplanet.com/tutorials/article.php/3468771.
[15] R. Gummadi, D. Wetherall, B. Greenstein, and S. Seshan. Understanding and Mitigating the Impact of RF Interference on 802.11 Networks. In *SIGCOMM*, 2007.
[16] D. Halperin, J. Ammer, T. Anderson, and D. Wetherall. Interference Cancellation: Better Receivers for a New Wireless MAC. In *Hotnets*, 2007.
[17] J. Hou, J. Smee, H. D. Pfister, and S. Tomasin. Implementing Interference Cancellation to Increase the EV-DO Rev A Reverse Link Capacity. *IEEE Communication Magazine*, 2006.
[18] E. Inc. Universal software radio peripheral. http://ettus.com.
[19] G. Judd and P. Steenkiste. Using Emulation to Understand and Improve Wireless Networks and Applications. In *NSDI*, 2005.
[20] P. Karn. MACA–A New Channel Access Method for packet Radio. *9th Computer Networking Conf.*, 1990.
[21] S. Katti, S. Gollakota, and D. Katabi. Embracing Wireless Interference: Analog Network Coding. In *ACM SIGCOMM 2007*.
[22] S. Khurana, A. Kahol, and A. P. Jayasumana. Effect of Hidden Terminals on the Performance of IEEE 802.11 MAC Protocol, 1998.
[23] A. Kochut, A. Vasan, A. Shankar, and A. Agrawala. Sniffing out the correct Physical Layer Capture model in 802.11b, 2004.
[24] E. A. Lee and D. G. Messerschmitt. *Digital Communications*. Boston: Kluwer Academic, 1988.
[25] J. Lee, W. Kim, S.-J. Lee, D. Jo, J. Ryu, T. Kwon, and Y. Choi. An Experimental Study on the Capture Effect in 802.11a Networks, 2007.
[26] Y. Li, L. Qiu, Y. Zhang, R. Mahajan, Z. Zhong, G. Deshpande, and E. Rozner. Effects of Interference on Wireless Mesh Networks: Pathologies and a Preliminary Solution. In *HotNets*, 2007.
[27] H. Meyr, M. Moeneclaey, and S. A. Fechtel. *Digital Communication Receivers: Synchronization, Channel Estimation, and Signal Processing*. John Wiley & Sons, 1998.
[28] A. Muqattash and M. Krunz. CDMA-Based MAC Protocol for Wireless Ad Hoc Networks. In *ACM MOBIHOC*, 2003.
[29] Netgear. Reference Manual for the NETGEAR ProSafe 802.11g Wireless Access point WG102.
[30] C. Reis, R. Mahajan, M. Rodrig, D. Wetherall, and J. Zahorjan. Measurement-Based Models of Delivery and Interference. In *SIGCOMM*, 2006.
[31] J. K. Tan. An Adaptive Orthogonal Frequency Division Multiplexing Baseband Modem for Wideband Wireless Channels. Master's thesis, MIT, 2006.
[32] D. Tse and P. Vishwanath. *Fundamentals of Wireless Communications*. Cambridge University Press, 2005.
[33] D. Tse, P. Viswanath, and L. Zheng. Diversity-Multiplexing Tradeoff in Multiple Access Channels. *IEEE Transaction on Information Theory*, pages 1859–74, 2004.
[34] S. Verdu. *Multiuser Detection*. Cambridge University Press, 1998.
[35] S. Verdu and S.Shamai. Spectral Efficiency of CDMA with Random Spreading. *IEEE Transaction on Information Theory*, pages 622–40, 1999.
[36] A. J. Viterbi. Very Low Rate Convolutional Codes for Maximum Theoretical Performance of Spread-Spectrum Multiple-Access Channels. *IEEE Journal on Selected Areas in Communications (JSAC)*, 8:641–649, May 1990.
[37] C. Ware, J. Judge, J. Chicharo, and E. Dutkiewicz. pages 159–163 vol.1.
[38] I. . WG. Wireless lan medium access control (mac) and physical layer (phy) specifications. *Standard Specification,IEEE, 1999*.
[39] G. Woo, P. Kheradpour, and D. Katabi. Beyond the Bits: Cooperative Packet Recovery Using PHY Information. In *ACM MobiCom*, 2007.
[40] K. Xu, M. Gerla, , and S. Bae. Effectiveness of RTS/CTS Handshake in IEEE 802.11 Based Ad Hoc Networks. *In Ad Hoc Network Journal*, 2003.
[41] J. Zhu, X. Guo, S. Roy, and K. Papagiannaki. CSMA Self-Adaptation based on Interference Differentiation. In *IEEE Globecom*, 2007.

APPENDIX

A. Proof of Lemma 7.1 Let us denote the duration of the slot time by S, ACK duration by ACK, SIFS duration by $SIFS$, and the initial congestion window by CW. We need the offset between the two colliding packets in the second collision to be greater than $SIFS+ACK$. Since in the second collision, Alice and Bob randomly pick a slot in the congestion window of size $2CW$, the probability that Alice picks a slot close enough to Bob to have an offset of less than SIFS+ACK is upper bounded by $\frac{SIFS+ACK}{CW.S}$. Thus the probability that the offset between the packets suffices to send an ACK is lower bounded by $1 - \frac{SIFS+ACK}{CW.S}$. For the backward-compatible 802.11g networks, the parameters are $S = 20\mu s$, $ACK = 30\mu s$, $SIFS = 10\mu s$ [13]. Substituting in the above equations, we find that the success probability is at least 0.9375.